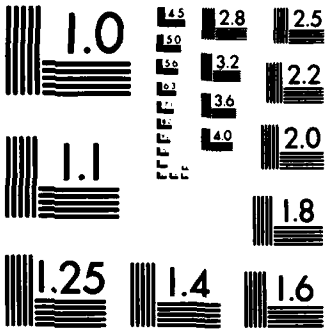
AD-A148 056   EGP (EXTERIOR GATEWAY PROTOCOL) GATEWAY UNDER BERKELEY   1/1
UNIX 42(U) UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL
REY INFORMATION S.. P KIRTON OCT 84 ISI/RR-84-145
UNCLASSIFIED   MDA903-81-C-0335   F/G 9/2   NL

END


MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

9

ISI Research Report
ISI/RR-84-145
October 1984

University of Southern California

Paul Kirton

EGP Gateway Under Berkeley UNIX 4.2

INFORMATION SCIENCES INSTITUTE ISI *213/822-1511*
4676 Admiralty Way/Marina del Rey/California 90292-6695

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ISI/RR-84-145	2. GOVT ACCESSION NO. AD-A148 056	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* EGP Gateway Under Berkeley UNIX 4.2		5. TYPE OF REPORT & PERIOD COVERED Research Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR*(s)* Paul Kirton		8. CONTRACT OR GRANT NUMBER*(s)* MDA903 81 C 0335
9. PERFORMING ORGANIZATION NAME AND ADDRESS USC/Information Sciences Institute 4676 Admiralty Way Marina del Rey, CA 90292-6695		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE October 1984
		13. NUMBER OF PAGES 43
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

This document is approved for public release and sale; distribution is unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

....

18. SUPPLEMENTARY NOTES

....

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

ARPANET, autonomous systems, computer networks, EGP, Exterior Gateway Protocol, gateways, internetwork communication, network management, protocols, routing algorithms, UNIX

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This report describes an implementation of the Exterior Gateway Protocol that runs under the UNIX 4.2 BSD operating system. Some issues related to local network configurations are also discussed. top. 1

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

ISI Research Report
ISI/RR-84-145
October 1984

University of Southern California

Paul Kirton

EGP Gateway Under Berkeley UNIX 4.2

INFORMATION SCIENCES INSTITUTE
213/822-1511
4676 Admiralty Way/Marina del Rey/California 90292-6695

This research is supported by the Defense Advanced Research Projects Agency under Contract No. MDA903 81 C 0335, and by a Telecom Australia Development Award. Views and conclusions contained in this report are the author's and should not be interpreted as representing the official opinion or policy of DARPA, the U.S. Government, Telecom Australia, or any person or agency connected with them.

CONTENTS

ACKNOWLEDGMENTS

I acknowledge with thanks the many people who have helped me with this project, but in particular, Dave Mills, who suggested the project, Jon Postel for discussion and encouragement, Liza Martin for providing the initial EGP code, Berkeley for providing the "routed" code, Mike Brescia for assistance with testing, Telecom Australia for funding me, and ISI for providing facilities.

1. INTRODUCTION

[illegible] 1673 The Exterior Gateway Protocol (EGP) [Rosen 82, Seamonson & Rosen 84, Mills 84a] has been specified to allow autonomous development of different gateway systems while still maintaining global distribution of internet routing information. EGP provides a means for different autonomous gateway systems to exchange information about the networks that are reachable via them.

This report mainly describes an implementation of EGP that runs as a user process under the Berkeley UNIX* 4.2 operating system run on a VAX** computer. Some related issues concerning local autonomous system configurations are also discussed.

The EGP implementation is experimental and is not a part of UNIX 4.2 BSD. It is anticipated that Berkeley will incorporate a version of EGP in the future.

The program is written in C. The EGP part is based on the C-Gateway code written by Liza Martin at MIT and the route management part is based on a UNIX 4.2 BSD route management daemon, "routed".

The EGP functions are consistent with the specification of [Mills 84a] except where noted.

A knowledge of EGP as described in [Seamonson & Rosen 84; Mills 84a] is assumed.

Requests for documentation and copies of the EGP program should be sent to Joyce Reynolds (JKReynolds@USC-ISIF.ARPA). Software support is not provided.

This chapter discusses motivation for the project, Chapter 2 describes the gateway design, Chapter 3 is on testing, Chapter 4 suggests some enhancements, Chapter 5 discusses topology issues, Chapter 6 discusses how to run the EGP program, and Chapter 7 describes the software. Chapters 1 to 5 are also published as RFC 911 [Kirton 84].

1.1 Motivation for Development

With the introduction of EGP, the internet gateways will be divided into a "core" autonomous system (AS) of gateways maintained by Bolt Beranek and Newman, Inc. (BBN) and many "stub" AS's that are maintained by different organizations and that have at least one network in common with a core AS gateway. The core AS will act as a hub for passing on routing information between different stub AS's so that it will only be necessary for stub AS's to conduct EGP with a core gateway. Further detail is given in [Rosen 82].

*UNIX is a trademark of AT&T.

**VAX is a trademark of Digital Equipment Corporation.

At the time of this project there were 28 "non-routing" gateways in the internet. Non-routing gateways did not exchange routing information but required static entries in the core gateway routing tables. Since August 1, 1984, these static entries have been eliminated and previously non-routing gateways are required to communicate this information to the core gateways dynamically via EGP [Postel 84].

At USC/Information Sciences Institute (ISI) there was a non-routing gateway to the University of California at Irvine network (UCI-ICS). With the elimination of non-routing gateways from the core gateway tables it is now necessary to inform the core ISI gateway of the route to UCI-ICS using EGP.

ISI has suggested building a backup gateway between ISI-NET and the ARPANET in case the core ISI gateway is down. Such a gateway would need to convey routing information via EGP. Details of the ISI network configuration are discussed in Section 5.2.

Of the 28 non-routing gateways, 23 were implemented by UNIX systems, including ISI's. Also, ISI's proposed backup gateway was a UNIX system. Thus there was a local and general need for an EGP implementation to run under UNIX. The current version of UNIX that included Department of Defense (DoD) protocols was Berkeley UNIX 4.2, so this was selected.

1.2 Overview of EGP

This report assumes a knowledge of EGP; however, a brief overview is given here for completeness. For further details refer to [Rosen 82] for the background to EGP, [Seamonson & Rosen 84] for an informal description, and [Mills 84a] for a more formal specification and implementation details.

EGP is generally conducted between gateways in different AS's that share a common network, that is, neighbor gateways. EGP consists of three procedures: neighbor acquisition, neighbor reachability, and network reachability.

Neighbor acquisition is a two-way handshake in which gateways agree to conduct EGP by exchanging Request and Confirm messages which include the minimum Hello and Poll intervals. Acquisition is terminated by an exchange of Cease and Cease-ack messages.

Neighbor reachability is a periodic exchange of Hello commands and I-H-U (I heard you) responses to ensure that each gateway is up. Currently a 30-second minimum interval is used across the ARPANET. Only one gateway need send commands, as the other can use them to determine reachability. A gateway sending reachability commands is said to be in the active mode, while a gateway that just responds is in the passive mode.

Network reachability is determined by an exchange of Poll commands and Update responses which indicate the networks reachable via one or more gateways on the shared network. Currently a two-minute minimum interval is used across the ARPANET.

2. GATEWAY DESIGN

EGP is a polling protocol with loose timing constraints. Thus the only gateway function requiring good performance is packet forwarding. UNIX 4.2 already has packet forwarding built into the kernel where best performance can be achieved. At the time of writing, UNIX 4.2 does not send ICMP (Internet Control Message Protocol) redirect messages for misrouted packets. This is a requirement of internet gateways and will later be added by Berkeley.

The EGP and route update functions are implemented as a user process. This facilitates development and distribution as only minor changes need to be made to the UNIX kernel. This is a similar approach to the UNIX route distribution program "routed" [Berkeley 83], which is based on the Xerox NS Routing Information Protocol [Xerox 81].

2.1 Routing Tables

A route consists of a destination network number, the address of the next gateway to use on a directly connected network, and a metric giving the distance in gateway hops to the destination network.

There are two sets of routing tables, the kernel tables (used for packet forwarding) and the EGP process tables. The kernel has separate tables for host and network destinations. The EGP process maintains only the network routing tables. The EGP tables are updated when EGP Update messages are received. When a route is changed the kernel network tables are updated via the SIOCADDRT and SIOCDELRT ioctl system calls. At initialization the kernel network routing tables are read via the kernel memory image file, /dev/kmem, and copied into the EGP tables for consistency.

This EGP implementation is designed to run on a gateway that is also a host. Because of the relatively slow polling to obtain route updates, it is possible that the host may receive notification of routing changes via ICMP redirects before the EGP process is notified via EGP. Redirects update the kernel tables directly. The EGP process listens for redirect messages on a raw socket and updates its routing tables to keep them consistent with the kernel.

The EGP process routing tables are maintained as two separate tables, one for exterior routes (via different AS gateways) and one for interior routes (via the gateways of this AS). The exterior routing table is updated by EGP Update messages. The interior routing table is currently static and is set at initialization time. It includes all directly attached nets, determined by the SIOCGIFCONF ioctl system call and any interior non-routing gateways read from the EGP initialization file, EGPINITFILE (see Section 6.3). The interior routing table could in future be updated dynamically by an Interior Gateway Protocol (IGP).

Maintaining separate tables for exterior and interior routing facilitates the preparation of outgoing Update messages which only contain interior routing information [Mills 84b]. It also permits alternative external routes to the internal routes to be saved as a backup in case an interior route fails. Alternate routes are flagged, RTS_NOTINSTALL, to indicate that the kernel routes should not be updated. In the current implementation alternate routes are not used.

2.1.1 Incoming Updates

EGP Updates are used to update the exterior routing table if one of the following is satisfied:

- No routing table entry exists for the destination network and the metric indicates the route is reachable (< 255).
- The advised gateway is the same as the current route.
- The advised distance metric is less than the current metric.
- The current route is older (plus a margin) than the maximum poll interval for all acquired EGP neighbors. That is, the route was omitted from the last Update.

If any exterior route entry, except the default route, is not updated by EGP within four minutes or three times the maximum poll interval, whichever is the greater, it is deleted.

If there is more than one acquired EGP neighbor, the Update messages received from each are treated the same way in the order they are received.

In the worst case, when a route is changed to a longer route and the old route is not first notified as unreachable, it could take two poll intervals to update a route. With the current poll interval this could be four minutes. Under UNIX 4.2 BSD, TCP connections (Transmission Control Protocol) are closed automatically after they are idle for six minutes, so this worst case does not result in the automatic closure of TCP connections.

2.1.2 Outgoing Updates

Outgoing Updates include the direct and static networks from the interior routing table, except for the network shared with the EGP neighbor. The networks that are allowed to be advised in Updates may be specified at initialization in EGPINITFILE. This allows particular routes to be excluded from exterior updates in cases where routing loops could be a problem. This option is also necessary when there is a non-routing gateway belonging to a different AS which has not yet implemented EGP. Its routes may need to be included in the kernel routing table, but they are not allowed to be advised in outgoing updates.

If the interior routing table includes other interior gateways on the network shared with the EGP neighbor they are included in Updates as the appropriate first hop to their attached networks.

The distance to networks is set as in the interior routing table except if the route is marked down, in which case the distance is set to 255. At present, routes are marked down only if the outgoing interface is down. The state of all interfaces is checked prior to preparing each outgoing Update using the SIOCGIFFLAGS ioctl system call.

Unsolicited Updates are not sent.

2.2 Neighbor Acquisition

EGPINITFILE lists the addresses of trusted EGP neighbor gateways, which are read at initialization. These will usually be core gateways, as only core gateways provide full internet routing information. At the time of writing three core gateways on the ARPANET support EGP: CSS-GATEWAY, ISI-GATEWAY, and PURDUE-CS-GW. Two core gateways on the MILNET support EGP: BBN-MINET-A-GW and AERONET-GW.

EGPINITFILE also includes the maximum number of these gateways that should be acquired at any one time. This is usually expected to be just one. If this gateway is declared down, another gateway on the list will then be acquired automatically in sufficient time to ensure that the current routes are not timed out.

The gateway will accept acquisitions only from neighbors on the trusted list and will not accept them if it already has acquired its maximum quota. This prevents Updates being accepted from possibly unreliable sources.

The ability to acquire core gateways that are not on the trusted list but that have been learned of indirectly via Update messages is not included, because not all core gateways run EGP.

New acquisition Requests are sent to neighbors in the order they appear in EGPINITFILE. No more new Requests than the maximum number of neighbors yet to be acquired are sent at once. Any number of outstanding Requests are retransmitted at 32-second intervals up to 5 retransmissions each, at which time the acquisition retransmission interval is increased to 4 minutes. Once the maximum number of neighbors has been acquired, unacquired neighbors with outstanding Requests are sent Ceases. This approach provides a compromise between fast response when neighbors do not initially respond and a desire to minimize the chance that a neighbor may be Ceased after it has sent a Confirm but before it has been received. If the specified maximum number of neighbors cannot be acquired, Requests are retransmitted indefinitely to all unacquired neighbors.

2.3 Hello and Poll Intervals

The Request and Confirm messages include minimum values for Hello and Poll intervals. The advised minimums are 30 and 120 seconds, respectively. These are the values currently used by both the core gateways and this gateway.

The received intervals are checked against upper bounds to guard against nonsense values. The upper bounds are currently set at 120 and 480 seconds, respectively. If they are exceeded by any neighbor, that particular neighbor is considered bad and is not sent further Requests for one hour. This allows the situation to be corrected at the other gateway and normal operation to automatically resume from this gateway without unnecessary network traffic.

The actual Hello and Poll intervals are computed by first selecting the maximum of the intervals advised by this gateway and its peer. A 2-second margin is then added to the Hello interval to account for possible network delay variations, and the Poll interval is increased to the next integer ratio of the Hello interval. This results in 32-second Hello and 128-second Poll intervals.

If an Update is not received in response to a Poll, at most one repoll (same sequence number) is sent instead of the next scheduled Hello.

2.4 Neighbor Cease

If the EGP process is sent a SIGTERM signal via the Kill command, all acquired neighbors are sent Cease(going down) commands. Ceases are retransmitted at the hello interval at most three times. Once all have either responded with Cease-acks or been sent three retransmitted Ceases, the process is terminated.

2.5 Neighbor Reachability

Only active reachability determination is implemented. It is done as recommended in [Mills 84a] with a minor variation noted below.

A shift register of responses is maintained. For each Poll or Hello command sent, a zero is shifted into the shift register. If a response (I-H-U, Update, or Error) is received with the correct sequence number, the zero is replaced by a one. Before each new command is sent the reachability is determined by examining the last four entries of the shift register. If the neighbor was considered reachable and at most one response was received, the neighbor is now considered unreachable. If the neighbor was considered unreachable and at least three responses were received, it is now considered reachable.

A neighbor is considered reachable immediately after acquisition, so that the first poll received from a core gateway (once it considers this gateway reachable) will be responded to with an Update. Polls are not sent unless a neighbor is considered reachable and it has not advised that it considers this gateway unreachable in its last Hello, I-H-U, or Poll message. This prevents the first Poll from being discarded after a down/up transition. This constraint is important, as the Polls are used for reachability determination. Following acquisition, at least one message must be received before the first Poll is sent. This is to determine that the peer does not consider this gateway down. Because of this requirement, at least one Hello must usually be sent prior to the first Poll. The discussion of this paragraph differs from [Mills 84a], which recommends that a peer be considered down following acquisition and allows Polls to be sent as soon as the peer is considered up. This is the only significant departure from the recommendations in [Mills 84a].

Polls received by peers that are considered unreachable are sent an Error response, which allows their reachability determination to progress correctly. This action is an option within [Mills 84a].

When a neighbor becomes unreachable, all routes using it as a gateway are deleted from the routing table. If there are known unacquired neighbors, the unreachable gateway is ceased and an attempt is made to acquire a new neighbor. If all known neighbors are acquired, the reachability determination is continued for 30 minutes ([Mills 84a] suggests 60 minutes), after which time the unreachable neighbor is ceased and reacquisition is attempted every 4 minutes. This is aimed at reducing unnecessary network traffic.

If valid Update responses are not received for three successive polls, the neighbor is ceased and an alternative is acquired or reacquisition is attempted in four minutes. This provision is included in case erroneous Update data formats are being sent by the neighbor. This situation did occur on one occasion during testing.

2.6 Sequence Numbers

Sequence numbers are managed as recommended in [Mills 84a]. Single send and receive sequence numbers are maintained for each neighbor. The send sequence number is initialized to zero and is incremented before each new Poll (not repoll) is sent and at no other time. The send sequence number is used in all commands. The receive sequence number is maintained by copying the sequence number of the last Request, Hello, or Poll command received from a neighbor. This sequence number is used in outgoing Updates. All responses (including Error responses) return the sequence number of the message just received.

2.7 Treatment of Excess Commands

If more than 20 commands are received from a neighbor in any 8-minute period, the neighbor is considered bad, Ceased, and reacquisition prevented for one hour.

At most one repoll (same sequence number) received before the poll interval has expired (less a four-second margin for network delay variability) is responded to with an Update; others are sent an Error response. When an Update is sent in response to a repoll, the unsolicited bit is not set; this differs from the recommendation in [Mills 84a].

2.8 Inappropriate Messages

If a Confirm, Hello, I-H-U, Poll, or Update is received from any gateway (known or unknown) that is in the unacquired state, synchronization has probably been lost for some reason. A Cease(protocol violation) message is sent to try and reduce unnecessary network traffic. This action is an option in [Mills 84a].

2.9 Default Gateway

A default gateway may be specified in EGPINITFILE. The default route (net 0 in UNIX 4.2 BSD) is used by the kernel packet forwarder if there is no specific route for the destination network. This default route provides a final level of backup if all known EGP neighbors are unreachable. The default gateway is especially useful if there is only one available EGP neighbor, as in the ISI case (see Section 5.2.2).

The default route is installed at initialization and deleted after a valid EGP Update message is received. It is reinstalled if all known neighbors are acquired but none are reachable, if routes time out while there are no EGP neighbors that are acquired and reachable, and prior to process termination.

It is deleted after a valid EGP Update message is received, because the default gateway will not know any more routing information than can be learned via EGP. If it were not deleted, all traffic to unreachable nets would be sent to the default gateway under UNIX 4.2 forwarding strategy.

The default gateway should normally be set to a full-routing core gateway other than the known EGP neighbor gateways to give another backup in case all of the EGP gateways are down simultaneously.

3. TESTING

A few interesting cases that occurred during testing are briefly described in this section.

The use of sequence numbers was interpreted differently by different implementers. Consequently, some implementations rejected messages as having incorrect sequence numbers, resulting in the peer gateway being declared down. The main problem was that the specification was solely in narrative form which is prone to inconsistencies, ambiguities, and incompleteness. The more formal specification of [Mills 84a] has eliminated these ambiguities.

When testing the response to packets addressed to a neighbor gateway's interface that was not on the shared net, a loop resulted as both gateways repeatedly exchanged error messages indicating an invalid interface. The problem was that both gateways were sending Error responses after checking the addresses but before checking the EGP message type. This problem was rectified by not sending an Error response unless it was certain that the message was not itself an Error response.

On one occasion a core gateway had some form of data error in the Update messages which caused them to be rejected even though reachability was being satisfactorily conducted. This resulted in all routes being timed out. The solution was to count the number of successive Polls that did not result in valid Updates being received and, if this number reached three, to Cease EGP and attempt to acquire an alternative gateway.

Another interesting idiosyncrasy, reported by Mike Karels at Berkeley, results from having multiple gateways between the MILNET and the ARPANET. Each ARPANET host has an assigned gateway to use for access to the MILNET. In cases where the EGP gateway is a host as well as a gateway, the EGP Update messages may indicate a different MILNET/ARPANET gateway from the assigned one. When the host/gateway originates a packet that is routed via the EGP-reported gateway, it will receive a redirect to its assigned gateway. Thus the MILNET gateway can keep being switched between the gateway reported by EGP and the assigned gateway. A similar situation occurs when using routes to other nets reached via MILNET/ARPANET gateways.

4. FUTURE ENHANCEMENTS

4.1 Multiple Autonomous Systems

The present method of acquiring a maximum number of EGP neighbors from a trusted list implies that all the neighbors are in the same AS. The intention is that they all be members of the core AS. When the routing tables are updated, Updates are treated independently with no distinction made as to whether the advised routes are internal or external to the peer's AS. Also, routing metrics are compared without reference to the AS of the source.

If EGP is to be conducted with additional AS's besides the core AS, all neighbors on the list would need to be acquired in order to ensure that gateways from both AS's were always acquired. This would result in an unnecessary excess of EGP traffic if redundant neighbors were acquired for reliability. A more desirable approach would be to have separate lists of trusted EGP gateways and the maximum number to be acquired for each AS. Routing entries would need to have the source AS added so that preference could be given to information received from the owning AS (see Section 5.1.2).

4.2 Interface Monitoring

At present, interface status is only checked immediately prior to the sending of an Update in response to a Poll. The interface status could be monitored more regularly and an unsolicited Update sent when a change is detected. This is one area where the slow response of EGP polling could be improved. This is of particular interest to networks that may be connected by dial-in lines. When such a network dials in, its associated interface will be marked as up but it will not be able to receive packets until the change has been propagated by EGP. This case would be helped by the unsolicited Update message, but there would still be a delay while other non-core gateways polled core EGP gateways for the new routing information.

It was initially thought that a kernel EGP implementation might help in this case. However, the kernel does not presently pass interface status changes by interrupts, so a new facility would need to be incorporated. If this were done it might be just as easy to provide a user level signal when an interface status changed.

4.3 Network Level Status Information

At present, network level status reports, such as IMP Destination Unreachable messages, are not used to detect changes in the reachability of EGP neighbors or other neighbor gateways. This information should be used to improve the response time to changes.

4.4 Interior Gateway Protocol Interface

At present, any routing information that is interior to the AS is static and is read from the initialization file. The internal route management functions have been written so that it should be reasonably easy to interface an IGP for dynamic interior route updates. This is facilitated by the separation of the exterior and interior routing tables.

The outgoing EGP Updates will be correctly prepared from the interior routing table by rt_NRnets() whether static or dynamic interior routing is done. Functions are also provided for looking up, adding, changing, and deleting internal routes, i.e., rt_int_lookup(), rt_add(), rt_change(), and rt_delete(), respectively.

The interaction of an IGP with the current data structures basically involves three functions: updating the interior routing table using a function similar to rt_NRupdate(), preparing outgoing interior updates similarly to rt_NRnets(), and timing out interior routes similarly to rt_time().

5. TOPOLOGY ISSUES

5.1 Topology Restrictions and Routing Loops

5.1.1 Background

EGP is not a routing algorithm; it merely enables exterior neighbors to exchange routing information which is likely to to be needed by a routing algorithm. It does not pass sufficient information to prevent routing loops if cycles exist in the topology [Rosen 82].

Routing loops can occur when two gateways think there are alternate routes to reach a third gateway via each other. When the third gateway goes down they end up pointing to each other, forming a routing loop. Within the present core system, loops are broken by counting to "infinity" (the internet diameter in gateway hops). This (usually) works satisfactorily, because GGP propagates changes fairly quickly as routing updates are sent as soon as changes occur. Also, the diameter of the internet is quite small (5) and a universal distance metric, hop count, is used. However, this will be changed in the future.

With EGP, changes are propagated slowly. Although a single unsolicited NR message can be sent, it won't necessarily be passed straight on to other gateways who must hear about it indirectly. Also, the distance metrics of different AS's are quite independent and hence can't be used to count to infinity.

The initial proposal for preventing routing loops involves restricting the topology of AS's to a tree structure so that there are no multiple routes via alternate AS's. Multiple routes within the same AS are allowed as it is the interior routing strategies' responsibility to control loops.

[Mills 84b] has noted that even with the tree topology restriction, "we must assume that transient loops may form within the core system from time to time and that this information may escape to other systems; however, it would be expected that these loops would not persist for very long and would be broken in a short time within the core system itself. Thus a loop between non-core systems can persist until the first round of Update messages sent to the other systems after all traces of the loop have been purged from the core system or until the reachability information ages out of the tables, whichever occurs first".

With the initial simple stub EGP systems, the tree topology restriction could be satisfied. However, this does not provide sufficient robustness for the long term.

[Mills 83] proposed a procedure by which the AS's can dynamically reconfigure themselves such that the topology restriction is always met, without the need for a single "core" AS. One AS would own a shared net and its neighbor AS's would just conduct EGP with the owner. The owner would pass on such information indirectly as the core system does now. If the owning AS is defined to be closest to the root of the tree topology, any haphazard interconnection can form itself into an appropriate tree-structured routing topology. By routing topology I mean the topology as advised in routing updates. There may well be other physical connections, but if they are not advised they will not be used for routing. Each AS can conduct EGP with at most one AS that owns one of its shared nets. Any AS that is not conducting EGP over any net owned by another AS is the root of a subtree. It may conduct EGP with just one other AS that owns one of its shared nets. This "attachment" combines the two subtrees into a single subtree such that the overall topology is still a tree. Topology violations can be determined because two different AS's will report that they can reach the same net.

With such a dynamic tree, there may be preferred and backup links. In such cases it is necessary to monitor the failed link so that routing can be changed back to the preferred link when service is restored.

Another aspect to consider is the possibility of detecting routing loops and then breaking them. Expiration of the packet time-to-live (TTL) could be used to do this. If such a loop is suspected a diagnostic packet, such as ICMP echo, could be sent over the suspect route to confirm whether it is a loop. If a loop is detected, a special routing packet could be sent over the route that instructs each gateway to delete the route after forwarding the packet on. The acceptance of new routing information may need to be delayed for a hold down period. This approach would require sensible selection of the initial TTL. However, this is not done by many hosts.

5.1.2 Current Policy

Considering the general trend to increased network interconnection and the availability of alternative long-haul networks such as ARPANET, WBNET (wideband satellite network), and public data networks, the tree topology restriction is generally unacceptable. A less restrictive topology is currently recommended. The following is taken from [Mills 84b].

EGP topological model:

- An autonomous system consists of a set of gateways connected by networks. Each gateway *in the system must be reachable from every other gateway in its system* by paths including only gateways in that system.
- A gateway in a system may run EGP with a gateway in any other system as long as the path over which EGP itself is run does not include a gateway in a third system.
- The "core system" is distinguished from the others by the fact that only it is allowed to distribute reachability information about systems other than itself.
- At least one gateway in every system must have a net in common with a gateway in the core system.
- There are no topological or connectivity restrictions other than those implied above.

A gateway will use information derived from its configuration (directly connected nets), the IGP of its system, called S in the following (interior nets), and EGP (interior and exterior nets of neighboring systems) to construct its routing tables. If conflicts with respect to a particular net N occur, they will be resolved as follows:

- If N is directly connected to the gateway, all IGP and EGP reports about N are disregarded.
- If N is reported by IGP as interior to S and by EGP as either interior or exterior to another system, the IGP report takes precedence.
- If N is reported by EGP as interior to one system and exterior to another, the interior report takes precedence.
- If N is reported as interior by two or more gateways of the same system using EGP, the reports specifying the smallest hop count take precedence.
- In all other cases the latest received report takes precedence.

Old information will be aged from the tables.

The interim model provides an acceptable degree of self-organization. Transient routing loops can occur between systems, but these are eventually broken by old reachability information being aged out of the tables. Given the fact that transient loops can occur due to temporary core-system loops, the additional loops that might occur in the case of local nets homed to multiple systems does not seem to increase the risk significantly.

5.2 Present ISI Configuration

A simplified version of the ISI network configuration is shown in Figure 5-1. ISI-Hobgoblin can provide a backup gateway function to the core ISI-Gateway between the ARPANET and ISI-NET. ISI-Hobgoblin is a VAX 11/750 which runs Berkeley UNIX 4.2. The EGP implementation described in this report is run on ISI-Hobgoblin.

ISI-Troll is part of a split gateway to the University of California at Irvine network (UCI-ICS). The complete logical gateway consists of ISI-Troll, the 9600 baud link, and UCI-750A [Rose 84]. ISI-Troll runs Berkeley UNIX 4.1a and hence cannot run the EGP program. It is therefore a non-routing gateway. The existence of the UCI-ICS net must be advised to the core AS by ISI-Hobgoblin. This can be done by including an appropriate entry in the EGPINITFILE.

Hosts on ISI-NET, including ISI-Troll, have static route entries indicating ISI-Gateway as the first hop for all networks other than UCI-ICS and ISI-NET.

EGP can be conducted with ISI-Gateway across either the ARPANET or ISI-NET.

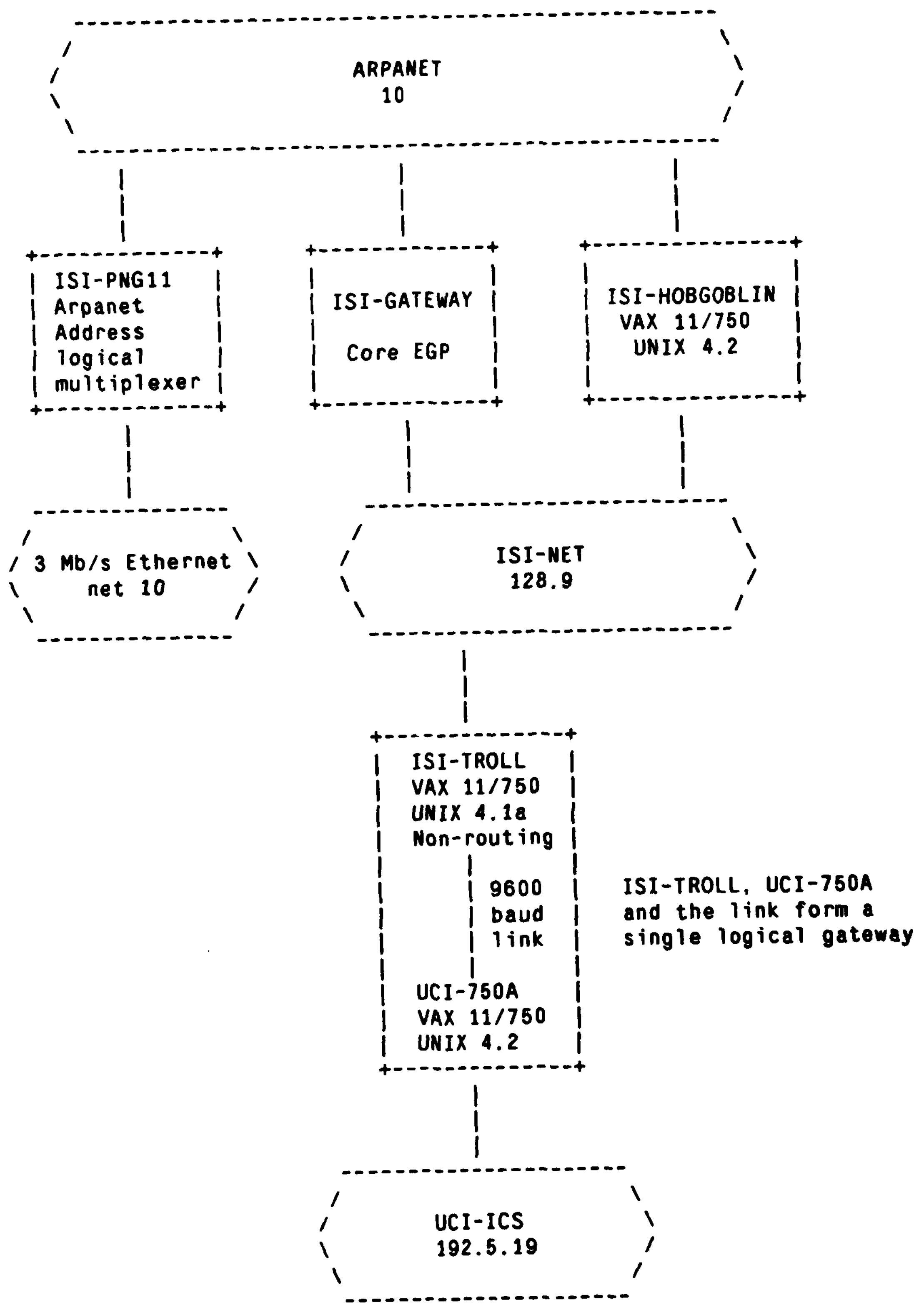


Figure 5-1: Simplified ISI network configuration

5.2.1 EGP Across the ARPANET

ISI-Hobgoblin will advise ISI-Gateway across the ARPANET, and hence the core system, that it can reach ISI-NET and UCI-ICS.

Packets from AS's exterior to ISI and destined for UCI-ICS will be routed via ISI-Gateway, ISI-Hobgoblin, and ISI-Troll. The extra hop via ISI-Gateway (or other core EGP gateway) is necessary because the core gateways do not currently pass on indirect-neighbor exterior gateway addresses in their IGP messages (Gateway-to-Gateway Protocol). Packets originating from UCI-ICS destined for exterior AS's will be routed via ISI-Troll and ISI-Gateway. Thus the incoming and outgoing packet routes are different.

Packets originating from ISI-Hobgoblin as a host and destined for exterior AS's will be routed via the appropriate gateway on the ARPANET.

UCI-ICS can communicate with exterior AS's only if ISI-Troll, ISI-Hobgoblin, and ISI-Gateway are all up. The dependence on ISI-Gateway could be eliminated if ISI-Troll routed packets via ISI-Hobgoblin rather than ISI-Gateway. However, as ISI-Hobgoblin is primarily a host and not a gateway, it is preferable that ISI-Gateway route packets when possible.

ISI-Hobgoblin can provide a backup gateway function to ISI-Gateway as it can automatically switch to an alternative core EGP peer if ISI-Gateway goes down. Even though ISI-Hobgoblin normally advises the core system that it can reach ISI-NET, the core uses its own internal route via ISI-Gateway in preference. For hosts on ISI-NET to correctly route outgoing packets they need their static gateway entries changed from ISI-Gateway to ISI-Hobgoblin. At present this would have to be done manually. This would only be appropriate if ISI Gateway were going to be down for an extended period.

5.2.2 EGP Across ISI-NET

ISI-Hobgoblin will advise ISI-Gateway across ISI-NET that its indirect neighbor, ISI-Troll, can reach UCI-ICS

All exterior packet routing for UCI-ICS will be via ISI-Gateway in both directions with no hops via ISI-Hobgoblin. Packets originating from ISI-Hobgoblin as a host and destined for exterior AS's will be routed via ISI-Gateway rather than the ARPANET interface, in both directions, thus taking an additional hop.

UCI-ICS can communicate with exterior AS's only if ISI-Troll and ISI-Gateway are up and ISI-Hobgoblin has advised ISI-Gateway of the UCI-ICS route. If ISI-Hobgoblin goes down, communication will still be possible because ISI-gateway (and other core gateways) do not time out routes to indirect neighbors. If ISI-Gateway then goes down, it will need to be readvised by ISI-Hobgoblin of the UCI-ICS route when it comes up.

Conducting EGP over ISI-NET rather than the ARPANET should provide more reliable service for UCI-ICS for the following reasons: ISI-Gateway is specifically designed as a gateway, it is expected to be up more than ISI-Hobgoblin, it is desirable to eliminate extra routing hops where possible, and the exterior routing information will persist after ISI-Hobgoblin goes down. If ISI-Hobgoblin is to be used in its backup mode, EGP can be restarted across the ARPANET after the new gateway routes are manually installed in the hosts. Therefore, EGP across ISI-NET was selected as the preferred mode of operation.

5.2.3 Potential Routing Loop

Because both ISI-Gateway and ISI-Hobgoblin provide routes between the ARPANET and ISI-NET, there is a potential routing loop. This topology in fact violates the original tree structure restriction Provided ISI-Hobgoblin does not conduct EGP simultaneously with ISI-Gateway over ISI-NET and the ARPANET, the gateways will know about the alternative route only from the shared EGP network and not from the other network. Thus a loop cannot occur. For instance, if EGP is conducted over ISI-NET, both ISI-Gateway and ISI-Hobgoblin will know about the alternative routes via each other to the ARPANET from ISI-NET, but they will not know about the gateway addresses on the ARPANET needed to access ISI-NET from the ARPANET. Thus they have insufficient routing data to be able to route packets in a loop between themselves.

5.3 Possible Future Configuration

5.3.1 Gateway to UCI-ICS

An improvement in the reliability and performance of the service offered to UCI-ICS can be achieved by moving the UCI-ICS interface from ISI-Troll to ISI-Hobgoblin Reliability will improve because the connection will require only ISI-Hobgoblin and its ARPANET interface to be up, and performance will improve because the extra gateway hop will be eliminated.

This configuration will also allow EGP to be conducted across the ARPANET, giving access to the alternative core gateways running EGP. This will increase the chances of reliably acquiring an EGP neighbor at all times It will also eliminate the extra hop via ISI-Gateway for packets originating from ISI-Hobgoblin, as a host, and destined for exterior networks.

This configuration change will be made sometime in the future. It was not done initially because ISI-Hobgoblin was experimental and was down more often than ISI-Troll.

5.3.2 Dynamic Switch to Backup Gateway

It was noted in Section 5.2.1 that ISI-Hobgoblin can provide a backup gateway function to ISI-Gateway between the ARPANET and ISI-NET. Such backup gateways could become a common approach to providing increased reliability.

At present the change over to the backup gateway requires the new gateway route to be manually entered for hosts on ISI-NET. This section describes a possible method for achieving this changeover dynamically when the primary gateway goes down.

The aim is to be able to detect when the primary gateway is down, and to have all hosts on the local network change to the backup gateway with a minimum amount of additional network traffic. The hosts should revert back to the primary gateway when it comes up again.

The proposed method is for only the backup gateway to monitor the primary gateway status and for it to notify all hosts of the new gateway address when there is a change.

5.3.2.1 Usual Operation

The backup gateway runs a process which sends reachability-probe messages, such as ICMP echoes, to the primary gateway every 30 seconds and uses the responses to determine reachability as for EGP. If the primary gateway goes down, a "gateway-address message" indicating the backup gateway address is broadcast (or preferably multicast) to all hosts. When the primary gateway comes up, another gateway message indicating the primary gateway address is broadcast. These broadcasts should be done four times at 30-second intervals to avoid the need for acknowledgments and knowledge of host addresses.

Each host runs a process that listens for gateway-address messages. If a different gateway is advised it changes the default gateway entry to the new address.

5.3.2.2 Host Initialization

When a host comes up, the primary gateway could be down, so it needs to be able to determine that it should use the backup gateway. The host can read the address of the primary and backup gateways from a static initialization file. It then sets its default gateway as the primary gateway and sends a "gateway-request message" to the backup gateway requesting the current gateway address. The backup gateway responds with a gateway-address message. If no response is received, the gateway-request should be retransmitted three times at 30-second intervals. If no response is received, the backup gateway can be assumed down and the primary gateway retained as the default. Whenever the backup gateway comes up it broadcasts a gateway-address message.

Alternatively, a broadcast (or multicast) gateway-request message could be defined to which only gateways would respond. The backup gateway-address message needs to indicate that it is the backup gateway so that future requests need not be broadcast. Again, three retransmissions should be used. However, the primary gateway also needs to broadcast its address whenever it comes up.

5.3.2.3 When Both Primary and Backup are Down

If the primary gateway is down and the backup knows it is going down, it should broadcast gateway-address messages indicating the primary gateway in case the primary gateway comes up first. However, the backup could go down without warning and the primary come up before it. If the primary gateway broadcasts a gateway-address message when it comes up, there is no problem. Otherwise, while hosts are using the backup gateway they should send a gateway-request message every 10 minutes. If no response is received, the message should be retransmitted three times at 30-second intervals and, if still no response, the backup should be assumed down and the primary gateway reverted to.

Thus the only time hosts need to send messages periodically is when the primary gateway does not send gateway-address messages on coming up and the backup gateway is being used. In some cases, such as at ISI, the primary gateway is managed by a different organization and experimental features cannot be conveniently added.

5.3.2.4 UNIX 4.2 BSD

One difficulty with the above is that there is no standard method of specifying internet broadcast or multicast addresses. Multicast addressing is preferable, as only those participating need process the message (interfaces with hardware multicast detection are available). In the case of UNIX 4.2 BSD, an internet address with zero local address is assumed for the internet broadcast address. However, the general Internet Addressing policy is to use an all ones value to indicate a broadcast function.

On UNIX 4.2 BSD systems, both the gateway and host processes could be run at the user level so that kernel modifications would not be required.

A User Datagram Protocol (UDP) socket could be reserved for host-backup-gateway communication.

Super user access to raw sockets for sending and receiving ICMP Echo messages requires a minor modification to the internet-family protocol switch table, the same as for ICMP Redirects as discussed in Section 6.5.

6. RUNNING THE EGP PROCESS

6.1 Starting the EGP Process and Trace Facilities

The EGP User Process is named "egpup". The command for starting the EGP User Process is:

```
egpup  [ -t [ i ] [ e ] [ r ] [ p ] ]  [logfile]
```

There are four levels of diagnostic tracing that are set by command arguments: t alone turns on all trace levels; i turns on tracing for internal error messages; e turns on tracing for EGP errors and state changes; r turns on tracing for route changes; and p turns on tracing for all packets sent and received. The file in which tracing information is to be written (appended) is defined by "logfile"; if not supplied the controlling terminal is used. Fatal error messages are always logged. A sample of the trace messages is given in the appendix.

The process should not be run until after user level interface and route initialization commands have been executed. The EGP process should not normally be run concurrently with the UNIX 4.2 BSD route management daemon "routed". This is discussed in Section 6.6. Once egpup is started, any manual changes to the kernel routing tables via the /etc/route command will result in inconsistencies between the EGP process routing tables and the kernel routing tables. No periodic consistency check is currently done. A few minor changes to the UNIX 4.2 BSD kernel are required for egpup to function correctly. These are described in Section 6.5.

6.2 Process Termination

If the EGP process is sent a SIGTERM signal via the Kill command, all acquired neighbors are sent Cease(going down) commands and, once all have either responded with Cease-acks or been sent three retransmitted Ceases, the process is terminated. This allows the static initialization data to be changed or trace options to be changed.

6.3 Site-Dependent Initialization

The name of the initialization file is defined by EGPINITFILE in the egp source file "defs.h". It is currently defined as "etc-egp". EGPINITFILE may include the following site-dependent initialization data: autonomous system number, addresses of trusted EGP neighbors, the maximum number of these to acquire at any one time, static interior routing information (optional), networks to be advised in outgoing Updates (optional), and a default gateway (optional).

The information in the file is identified by keywords at the start of a line. Any text to the right of a # is a comment. The keywords (bold type) and data formats are as follows:

```
autonomoussystem value1
egpneighbor name1
egpmaxacquire value2
net name2 gateway name3 metric value3          # optional
egpnetsreachable name4 name4 .... name4        # optional
defaultgateway name5                           # optional
```

where:

value1	is the gateway autonomous system number.
names	are either symbolic as located in /etc/hosts or /etc/networks or internet addresses in dot notation.
name1	is an EGP neighbor gateway. Normally all neighbors are on the same net. Attempts to acquire neighbors are made in the order in which the neighbors are listed.
value2	is the maximum number of EGP neighbors to be acquired.
name2	is a net reachable via a non-routing gateway name3 (belonging to this AS) at distance value3 hops from this gateway.
name4	is a net whose reachability is allowed to be made public in EGP Network Reachability update messages. Normally, the nets to be advised will be direct nets (other than the net shared with the EGP peer) and nets via interior non-routing gateways. If an egpnetsreachable statement is included, all nets not declared by it will be excluded from update messages.
name5	is a default gateway. The default route (net 0 in UNIX 4.2 BSD) is installed at initialization and deleted after a valid EGP Update message is received. It is reinstalled if all EGP neighbors are acquired but none are reachable, if routes time out while no EGP neighbors are acquired and reachable, and prior to EGP process termination. The default gateway should normally be set to a full-routing core gateway other than the listed EGP neighbor gateways to give another backup in case all of the EGP gateways are down simultaneously.

An example initialization file that could be used at ISI is as follows:

```
autonomoussystem 4
egpneighbor isi-gateway                  # 10.3.0.27 - Core EGP on arpanet
egpneighbor 10.2.0.25                    # css-gateway - core EGP on arpanet
egpneighbor purdue-cs-gw                 # 10.2.0.37 - core EGP on arpanet
egpmaxacquire 1
net uciics gateway isi-troll metric 1    # U.C. Irvine Network
egpnetsreachable isi-net arpanet uciics
defaultgateway isi-milnet-gw             # 10.2.0.22 - core with no EGP on arpanet
```

At the time of writing, ISI-GATEWAY, CSS-GATEWAY, and PURDUE-CS-GW are the only operational core gateways running EGP over the ARPANET, and BBN-MINET-A-GW and AERONET-GW the only ones for the MILNET.

Before running EGP with operational EGP gateways it should be tested with the test gateway, BBN-TEST3, 10.3.0.63, which is isolated from the rest of the core system.

Before bringing up an EGP gateway, it is necessary to have an Autonomous System number allocated by Joyce Reynolds (JKReynolds@USC-ISIF.ARPA).

6.4 Compiler Switches

The source file "defs.h" defines three compiler switches at the start.

INSTALL	if 0, prevents routing changes from being installed in the kernel, normally 1.
SAMENET	if 1, will cause a fatal error if all EGP neighbors read from EGPINITFILE are not on the same network, normally 0.
ALLOWNONNEIGHBORS	if 1, allows EGP peers who are not on a common network with this gateway to conduct EGP for testing, normally 0. If 1, SAMENET should be 0 and DEFAULTIF (in "defs.h") should define the internet address (host specific) of the source interface to use for sending and receiving EGP messages.

6.5 UNIX Interface

The EGP process uses the raw socket interface for the Internet Protocol (IP) to send and receive EGP packets and to receive ICMP redirects. This interface currently lacks a facility for arbitrarily assigning a protocol number to a socket. An entry must be added for EGP (protocol #8) to the UNIX kernel internet-family protocol switch table, inetsw[], which is initialized in the UNIX source file "../netinet/in_proto.c". The entry should be the same as for the raw socket but with the protocol number replaced by IPPROTO_EGP, which should be defined as 8 in "../netinet/in.h":

```
{  SOCK_RAW,       PF_INET,        IPPROTO_EGP,    PR_ATOMIC|PR_ADDR,
   rip_input,      rip_output,     0,              0,
   raw_usreq,
   0,              0,              0,              0,
}
```

The raw socket interface does not currently support wildcard addressing, so separate sockets are opened for each interface for both EGP and ICMP packets.

ICMP Redirects update the kernel routing tables directly. Redirect messages are meant to be available from raw sockets, but the standard kernel does not give the super user raw-socket access to ICMP (protocol #1). The internet-family protocol switch table must be modified to allow this. In "../netinet/in_proto.c change the entry for ICMP to:

```
{  SOCK_RAW,       PF_INET,        IPPROTO_ICMP,   PR_ATOMIC|PR_ADDR,
   icmp_input,     rip_output,     0,              0,
   raw_usreq,
   0,              0,              0,              0,
}
```

Redirects are then received. If redirects were not received there would be no great problem. It merely means that the kernel routing tables would be updated ahead of the EGP tables, which must wait for the next EGP Update message.

It should be noted that EGP packets received by raw sockets include the IP header, whereas ICMP redirect packets do not.

System calls are used for determining the interface configuration (SIOCGIFCONF ioctl), determining interface status (SIOCGIFFLAGS ioctl), and altering the kernel routing tables (SIOCADDRT and SIOCDELRT ioctl system calls).

The UNIX kernel currently sets the Type of Service field randomly for raw IP output packets. The gateways complain about this, so this should be set to zero in the UNIX source file "../netinet/raw_ip.c", rip_output().

A bug in the UNIX raw packet code is that once a packet has been sent on a raw socket all subsequent packets use the same route. This doesn't really affect EGP, as all peers must be on a shared net. However, this bug can cause trouble during testing if you try to peer with a neighbor not on a shared net as well as a neighbor on the shared net. The bug can be fixed (courtesy of Robert Scheifler at MIT) by zeroing the old route pointer to allow a new route to be allocated. In UNIX source file "../net/raw_usrreq.c", raw_usrreq(), case PRU_SEND, replace:

```
        if( rp->rcb_route.ro_rt)
                RTFREE(rp->rcb_route.ro_rt);
```

by:

```
        if( rp->rcb_route.ro_rt) {
                RTFREE(rp->rcb_route.ro_rt);
                rp->rcb_route.ro_rt = NULL;}
```

A UNIX kernel routing bug, which does not affect EGP, was found while considering effects of adding dynamic route management. TCP connections continue to use old routes even after they have been marked as down, because the IP output code fails to check whether the route is up. To fix this bug, in the UNIX source file "../netinet/ip_output.c", check that the route is up before using it and, if it is not, free it and allocate a new route. Also, in the UNIX source file "../net/route.c", rtalloc(), check that any supplied route is up and, if not, allocate a new route.

6.6 EGP and Routed

The EGP process can be run concurrently with the UNIX 4.2 BSD route management daemon "routed", or any other route management process, only if they manage separate parts of the routing information. This is because there is no means at present of coordinating their respective route changes.

If "routed" manages any of the routes managed by EGP, the EGP routing tables will become inconsistent with the kernel routing tables, resulting in error returns for some SIOCADDRT and SIOCDELRT ioctl system calls.

The EGP process manages only the kernel network routing table, not the host routing table. It assumes that routes interior to its own AS are static and reads these from EGPINITFILE.

When EGP is initialized it first reads the kernel network routes. Except for the default route and static interior routes, these will all be timed out and deleted after six minutes if they are not updated by EGP.

If "routed" is to be used as an Interior Gateway Protocol to provide dynamic routing information for interior routes, the EGP code needs to be modified so that its interior routing table is appropriately updated. This involves combining the functions of "routed" and the EGP process into a single route management daemon. Berkeley plans to do this at some future stage.

7. SOFTWARE DESCRIPTION

7.1 Source Files

egp-notes	notes from this report on the EGP program operation and description.
etc-egp	site-dependent initialization file defined by EGPINITFILE in defs.h.
makefile	
include.h	system and EGP header files to be included.
defs.h	compiler switches and miscellaneous definitions.
ext.c	external variable definitions.
main.c	main function of the EGP user process: packet reception, ICMP handler, and timer interrupt handler.
init.c	initialization functions for interface data, sockets, EGP neighbors, and state tables.
rt_init.c	initialization functions for routing tables including interface routes, kernel routes, non-routing gateways, and routes to be advised in exterior EGP Update messages.
egp.h	various EGP definitions, packet formats, and state table.
egp_param.h	various EGP parameter definitions.
egp.c	EGP functions.
rt_egp.c	EGP route update processing and preparation.
rt_table.h	routing table data and parameter definitions.
rt_table.c	routing table management functions.
if.h	interface data definition.
if.c	interface checking functions.
trace_egp.h	definitions for tracing.
trace_egp.c	tracing functions for route changes, received packets, and EGP messages.

7.2 Software Overview

At start up, the controlling function, main(), first sets up trace options based on command arguments, calls functions to do initialization, and calls timeout(), which starts periodic interrupts and waits to receive incoming EGP or ICMP redirect packets.

When an EGP packet is received, egpin() is called to handle it. It in turn calls a separate function for each EGP command type, which sends the appropriate response. When an ICMP packet is received icmpin() is called.

The timer interrupt routine, timeout(), calls egpjob() to perform periodic EGP processing such as reachability determination, command sending, and retransmission. It in turn calls separate functions to format each command type. Timeout() also periodically calls rt_time() to increment route ages and delete routes when too old.

7.3 Main Data Structures

The main data structures are interface, egpngh, and rt_entry.

Structure "interface" stores information about a directly attached interface such as name, internet address, and bound sockets. The interface structures are in a singly linked list pointed to by external variable "ifnet". This is similar to the structure used in UNIX 4.2 BSD "routed".

Structure "egpngh" stores state information for an EGP neighbor. There is one such structure allocated at initialization for each of the trusted EGP neighbors read from EGPINITFILE. The egpngh structures are in a singly linked list pointed to by external variable "egpngh". The major states of a neighbor are ACQUIRED, ACQUIRE_SENT, NEIGHBOR, and CEASE_SENT. In the NEIGHBOR state there are three reachability substates according to whether the neighbor and this gateway both consider each other up (BOTH_UP), this gateway considers its neighbor down (NG_DOWN), or the neighbor has reported that it considers this gateway down (ME_DOWN).

Structure "rt_entry" stores information about one particular route, similarly to "routed". Only routes to networks are used.

The interior routes are stored in a doubly linked list pointed to by the external variable "rt_interior". A single list is used because there are typically only a few internal routes, 3 in ISI's case. The exterior routes are organized as a hashed array of 19 doubly linked lists, as in "routed". The hash function is the network number of the destination network, modulo 19. The hash array is named nethash[].

7.4 Functions

This section describes the major software functions and their dependencies. Not all functions are included in the discussion.

7.4.1 Controlling Functions

main() first sets tracing options based on starting command arguments. It then calls the initialization functions described in the next section, sets up signal handlers for termination (SIGTERM) and timer interrupt (SIGALRM) signals, calls timeout() to start periodic interrupt processing, and finally waits in a loop to receive incoming packets.

timeout() is called when the periodic interrupt timer expires. It calls egpjob() and/or rt_time() at the appropriate times. The interrupt timer interval is set as the smallest value of the route age interval (60 seconds) and the periodic command intervals for all the EGP neighbors (typically 32 seconds).

7.4.2 Initialization Functions

init_if() initializes the interface tables for the internet interfaces configured into the system.

init_sock() sets up raw sockets on each interface for sending and receiving EGP messages or receiving ICMP redirect messages. Sockets are established on all interfaces even when trusted neighbors are all on the same network so that EGP requests from all gateways will be appropriately responded to.

init_egpngh() reads EGPINITFILE for EGP AS number and EGP neighbor addresses.

init_egp() initializes the state tables for each EGP neighbor.

rt_init() initializes the routing table doubly linked lists.

rt_readkernel() reads the kernel routes and initializes the exterior routing tables consistent with the kernel tables.

rt_ifinit() adds routes to the interior routing table for all directly attached networks and deletes these from the exterior table.

rt_dumbinit() reads EGPINITFILE for non-routing gateways and a default gateway and then deletes these from the exterior table if present and installs them in the kernel if they were not previously read from the kernel.

rt_NRadvise_init() reads EGPINITFILE for any user specification of the networks allowed to be advised in outgoing update messages. All other interior routes are flagged RTS_NOTADVISENR.

7.4.3 EGP Functions

egpjob() is called periodically by timeout() after each timer interrupt. It checks each neighbor's state table in turn and decides what action to take based on the state and time elapsed. It initiates the sending of Acquisition Request, Hello, and Poll commands and the retransmission of Acquisition Request, Poll, and Cease commands. Appropriate functions are called to format each of these messages. In the NEIGHBOR state it determines the current reachability of the peer. After 480 seconds have elapsed it calls **egpchkcmd()** to check whether peer commands are being received at an excess rate.

egpin() handles received EGP packets. It first checks for fragmented packets, valid EGP checksum, version, and length. If any of these are in error the packet is discarded. If the source and destination addresses match those expected in the state tables the appropriate function is called according to the EGP message type. Otherwise the appropriate Refuse, Cease-ack, Error, or Cease message is returned. Received Error messages are merely logged with no other action taken.

egpacq() handles received Neighbor Acquisition messages: Request, Confirm, Refuse, Cease, and Cease-ack. It calls **egpsetint()** to validate and set Hello and Poll intervals, **egpstngh()** to initialize state tables when the state changes to ACQUIRED, and **egpstunacq()** when the state changes to UNACQUIRED. The last two functions call **egpstime()** to recompute the interrupt timer interval.

egphello(), **egppoll()**, and **egpnr()** handle received EGP Neighbor Reachability messages (Hello and I-H-U), Poll, and network reachability Updates, respectively. The function egpnr() calls **rt_NRupdate()** to update the routing tables as described in Section 2.1.1, and the function rt_NRupdate() calls **rt_default()** to delete the default route.

egpsacq(), **egpshello()**, **egpspoll()**, **egpsnr()**, and **egpserr()** format outgoing Neighbor Acquisition, Neighbor Reachability, Poll, network reachability Update, and Error messages, respectively. They call **egp_cksum()** to compute the checksum and **egp_send()** to send the message. The function egp_send() calls **if_withnet()** to determine which interface, and hence socket, has the required source address.

The function egpsnr() calls **if_check()** to check the status of all interfaces, **rt_ifupdate()** to update interior route status if any interfaces have changed state, and **rt_NRnets()** to format the network reachability part of the outgoing update in accord with Section 2.1.2.

egpallcease() is called when the kill signal, SIGTERM, is received. It ceases each acquired neighbor by calling **egpcease()**.

7.4.4 Routing Table Management Functions

rt_ext_lookup() and **rt_int_lookup()** look up a destination network in the exterior or interior routing tables, respectively.

rt_add(), **rt_change()**, and **rt_delete** add, change, or delete a route entry, respectively, to/from either the interior or exterior routing table.[***]

rt_default adds or deletes a default route entry to/from the kernel.

rt_time() is called approximately every minute by timeout() to increment the route ages. If a route has been updated since the last increment, its age is set to zero. If a route is older than 4 minutes or 3 times the maximum poll interval, whichever is the greater, it is deleted. If old routes are deleted, rt_check_default() is called.

rt_check_default() checks whether any of the EGP neighbors are acquired and reachable. If not, rt_default() is called to install the default route.

rt_unreach() deletes all routes that use a specified gateway from the exterior routing table.

rt_redirect() changes the routing table entry in response to an ICMP redirect message.

7.4.5 Tracing Functions

traceaction() records changes to the routing tables. It is similar to the "routed" function.

tracercv() traces a received packet. It calls **traceegp()** to trace EGP packets.

[***] The functions rt_ext_lookup(), rt_add(), rt_change(), and rt_delete() are based on "routed" code.

8. RESOURCE USE

The EGP process when run on a VAX has a typical virtual memory size of 100 Kbytes, which consists of the following:

31 K	Code
7 K	Static compiled data
18 K	Data allocated at run time prior to the start of main() for unknown purposes
24 K	Dynamic data allocated by malloc() for library file-input functions during initialization
14 K	Dynamic data allocated by malloc() for EGP User Process data; only 5K is actually used
6 K	Stack
100 K	

Typically, resident memory is 42 Kbytes, of which 26K is code and 16K is data and stack.

EGP functions average 31 ms processing time per command/response pair. Based on the Hello and Polling intervals quoted here and assuming both peers conduct active reachability determination, the total CPU utilization is approximately 0.2 percent. This does not include the packet forwarding function (done in the kernel), which is typically the most significant gateway load.

A. SAMPLE TRACE OUTPUT

File "egpout" logs trace and error messages for the EGP user process, "egpup".
Tracing levels turned on: i e r p

```
Start egpup at Wed Jul 25 17:00:34 1984

init_if: interface name: ec0, address 128.9.0.42
init_if: interface name: imp0, address 10.1.0.52
init_egpngh: EGP neighbor 10.3.0.27
init_egpngh: EGP neighbor 10.2.0.25
init_egpngh: egpmaxacquire = 1
rt_readkernel: Initial routes read from kernel (if any):
ADD EXT dst 0.0.0.0, router 10.3.0.27, metric 254, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.19.0, router 128.9.0.44, metric 254, flags UP|GATEWAY, state CHANGED
ADD EXT dst 4.0.0.0, router 10.1.0.20, metric 254, flags UP|GATEWAY, state CHANGED
ADD EXT dst 26.0.0.0, router 10.2.0.22, metric 254, flags UP|GATEWAY, state CHANGED
ADD EXT dst 128.4.0.0, router 10.0.0.111, metric 254, flags UP|GATEWAY, state CHANGED
rt_ifinit: interior routes for direct interfaces:
ADD INT dst 10.0.0.0, router 10.1.0.52, metric 0, flags UP, state INTERFACE|CHANGED
ADD INT dst 128.9.0.0, router 128.9.0.42, metric 0, flags UP, state INTERFACE|CHANGED

***Routes are being installed in kernel

rt_dumbinit: non-routing gateway routes (if any):
DELETE EXT dst 192.5.19.0, router 128.9.0.44, metric 254, flags UP|GATEWAY, state CHANGED
ADD INT dst 192.5.19.0, router 128.9.0.44, metric 1, flags UP|GATEWAY, state PASSIVE|CHANGED
CHANGE dst 0.0.0.0, router 10.2.0.22, metric 254, flags UP|GATEWAY, state CHANGED
rt_NRadvise_init: user specified nets allowed to be advised in NR updates:
 isi-net arpanet uciics
main: commence EGP route updates:

EGP SENT 10.1.0.52 -> 10.3.0.27 Wed Jul 25 17:00:46 1984
vers 2, typ 3, code 0, status 1, AS # 4, id 1, hello int 30, poll int 120

EGP RECV 10.3.0.27 -> 10.1.0.52 Wed Jul 25 17:00:47 1984
vers 2, typ 3, code 1, status 1, AS # 256, id 1, hello int 30, poll int 120
egpstngh: acquired 10.3.0.27 Wed Jul 25 17:00:47 1984

EGP RECV 10.3.0.27 -> 10.1.0.52 Wed Jul 25 17:00:47 1984
vers 2, typ 5, code 0, status 2, AS # 256, id 30137

EGP SENT 10.1.0.52 -> 10.3.0.27 Wed Jul 25 17:00:47 1984
vers 2, typ 5, code 1, status 1, AS # 4, id 30137

EGP SENT 10.1.0.52 -> 10.3.0.27 Wed Jul 25 17:01:19 1984
vers 2, typ 5, code 0, status 1, AS # 4, id 1

EGP RECV 10.3.0.27 -> 10.1.0.52 Wed Jul 25 17:01:19 1984
```

```
vers 2, typ 5, code 1, status 1, AS # 256, id 1

EGP RECV 10.3.0.27 -> 10.1.0.52 Wed Jul 25 17:01:21 1984
vers 2, typ 2, code 0, status 0, AS # 256, id 30142, src net 10.0.0.0

EGP SENT 10.1.0.52 -> 10.3.0.27 Wed Jul 25 17:01:21 1984
vers 2, typ 1, code 0, status 1, AS # 4, id 30142, src net 10.0.0.0, # int 1, # ext 0
1 0 52 2 0 1 128 9 1 1 192 5 19

EGP RECV 10.3.0.27 -> 10.1.0.52 Wed Jul 25 17:01:47 1984
vers 2, typ 5, code 0, status 1, AS # 256, id 30146

EGP SENT 10.1.0.52 -> 10.3.0.27 Wed Jul 25 17:01:47 1984
vers 2, typ 5, code 1, status 1, AS # 4, id 30146

EGP SENT 10.1.0.52 -> 10.3.0.27 Wed Jul 25 17:01:51 1984
vers 2, typ 2, code 0, status 1, AS # 4, id 2, src net 10.0.0.0

EGP RECV 10.3.0.27 -> 10.1.0.52 Wed Jul 25 17:01:51 1984
vers 2, typ 1, code 0, status 1, AS # 256, id 2, src net 10.0.0.0, # int 26, # ext 1
3 0 27 2 0 1 128 9 2 1 192 5 19 2 0 5 4 0 1 192 1 2 1 3 10 192 1 7 192 1 4 2 1 128 2 255 1 59
5 0 5 3 0 1 26 1 7 24 192 5 9 128 44 6 128 21 192 5 38 128 25 2 15 128 49 192 5 35 192 5 21 128 20
 192 5 25 192 5 22 128 3 128 8 192 5 15 192 5 27 192 5 26 192 5 52 192 5 66 192 5 54 192 5 24
1 0 20 4 0 2 4 7 1 4 192 5 28 192 5 29 32 128 16 2 2 192 5 30 18 3 3 128 31 192 10 41 128 52
2 0 22 1 2 1 192 5 89 2 0 25 2 1 3 128 39 192 5 46 192 5 31 2 1 192 5 56 1 0 28 1 0 1 192 5 18
2 0 37 2 0 1 128 10 2 1 192 5 48 1 0 49 2 0 2 192 1 3 128 11 2 1 192 5 51 4 0 51 1 2 2 192 5 14 192 5 64
3 0 72 2 0 1 8 2 1 192 5 88 0 0 94 1 0 1 192 5 2 0 0 4 1 1 1 192 5 12 2 0 9 1 1 1 128 36
1 0 11 1 1 1 36 0 0 15 1 1 1 192 5 37 0 0 25 1 1 1 192 5 11 7 0 49 1 1 1 192 5 58 1 0 54 1 1 1 192 5 7
2 0 78 1 1 1 128 32 3 0 89 1 1 1 192 5 43 3 0 91 1 1 1 192 5 8 3 0 96 1 1 1 192 5 36 0 0 68 1 2 1 192 5 6
5 0 51 1 0 1 128 18 5 0 63 2 0 1 14 2 1 128 42 0 0 111 1 0 2 128 5 128 4
ADD EXT dst 128.9.0.0, router 10.3.0.27, metric 0, flags UP|GATEWAY, state CHANGED|NOTINSTALL
ADD EXT dst 192.5.19.0, router 10.3.0.27, metric 2, flags UP|GATEWAY, state CHANGED|
                                                                                  NOTINSTALL
ADD EXT dst 192.1.2.0, router 10.2.0.5, metric 0, flags UP|GATEWAY, state CHANGED
ADD EXT dst 10.0.0.0, router 10.2.0.5, metric 1, flags UP|GATEWAY, state CHANGED|NOTINSTALL
ADD EXT dst 192.1.7.0, router 10.2.0.5, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.1.4.0, router 10.2.0.5, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 128.2.0.0, router 10.2.0.5, metric 2, flags UP|GATEWAY, state CHANGED
CHANGE dst 26.0.0.0, router 10.5.0.5, metric 0, flags UP|GATEWAY, state CHANGED
ADD EXT dst 24.0.0.0, router 10.5.0.5, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.9.0, router 10.5.0.5, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 128.44.0.0, router 10.5.0.5, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 6.0.0.0, router 10.5.0.5, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 128.21.0.0, router 10.5.0.5, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.38.0, router 10.5.0.5, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 128.25.0.0, router 10.5.0.5, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 128.49.0.0, router 10.5.0.5, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.35.0, router 10.5.0.5, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.21.0, router 10.5.0.5, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 128.20.0.0, router 10.5.0.5, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.25.0, router 10.5.0.5, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.22.0, router 10.5.0.5, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 128.3.0.0, router 10.5.0.5, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 128.8.0.0, router 10.5.0.5, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.15.0, router 10.5.0.5, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.27.0, router 10.5.0.5, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.26.0, router 10.5.0.5, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.52.0, router 10.5.0.5, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.66.0, router 10.5.0.5, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.54.0, router 10.5.0.5, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.24.0, router 10.5.0.5, metric 2, flags UP|GATEWAY, state CHANGED
CHANGE dst 4.0.0.0, router 10.1.0.20, metric 0, flags UP|GATEWAY, state CHANGED
ADD EXT dst 7.0.0.0, router 10.1.0.20, metric 0, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.28.0, router 10.1.0.20, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.29.0, router 10.1.0.20, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 32.0.0.0, router 10.1.0.20, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 128.16.0.0, router 10.1.0.20, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.30.0, router 10.1.0.20, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 18.0.0.0, router 10.1.0.20, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 128.31.0.0, router 10.1.0.20, metric 3, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.10.41.0, router 10.1.0.20, metric 3, flags UP|GATEWAY, state CHANGED
ADD EXT dst 128.52.0.0, router 10.1.0.20, metric 3, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.89.0, router 10.2.0.22, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 128.39.0.0, router 10.2.0.25, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.46.0, router 10.2.0.25, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.31.0, router 10.2.0.25, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.56.0, router 10.2.0.25, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.18.0, router 10.1.0.28, metric 0, flags UP|GATEWAY, state CHANGED
ADD EXT dst 128.10.0.0, router 10.2.0.37, metric 0, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.48.0, router 10.2.0.37, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.1.3.0, router 10.1.0.49, metric 0, flags UP|GATEWAY, state CHANGED
ADD EXT dst 128.11.0.0, router 10.1.0.49, metric 0, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.51.0, router 10.1.0.49, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.14.0, router 10.4.0.51, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.64.0, router 10.4.0.51, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 8.0.0.0, router 10.3.0.72, metric 0, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.88.0, router 10.3.0.72, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.2.0, router 10.0.0.94, metric 0, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.12.0, router 10.0.0.4, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 128.36.0.0, router 10.2.0.9, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 36.0.0.0, router 10.1.0.11, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.37.0, router 10.0.0.15, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.11.0, router 10.0.0.25, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.58.0, router 10.7.0.49, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.7.0, router 10.1.0.54, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 128.32.0.0, router 10.2.0.78, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.43.0, router 10.3.0.89, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.8.0, router 10.3.0.91, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.36.0, router 10.3.0.96, metric 1, flags UP|GATEWAY, state CHANGED
ADD EXT dst 192.5.6.0, router 10.0.0.68, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 128.18.0.0, router 10.5.0.51, metric 0, flags UP|GATEWAY, state CHANGED
ADD EXT dst 14.0.0.0, router 10.5.0.63, metric 0, flags UP|GATEWAY, state CHANGED
ADD EXT dst 128.42.0.0, router 10.5.0.63, metric 2, flags UP|GATEWAY, state CHANGED
ADD EXT dst 128.5.0.0, router 10.0.0.111, metric 0, flags UP|GATEWAY, state CHANGED
CHANGE dst 128.4.0.0, router 10.0.0.111, metric 0, flags UP|GATEWAY, state CHANGED
DELETE DEFAULT dst 0.0.0.0, router 10.2.0.22, metric 254, flags UP|GATEWAY, state PASSIVE
rt_NRupdate: above routes from 10.3.0.27 updated Wed Jul 25 17:01:51 1984

EGP RECV 10.3.0.27 -> 10.1.0.52 Wed Jul 25 17:02:21 1984
vers 2, typ 5, code 0, status 1, AS # 256, id 30151

EGP SENT 10.1.0.52 -> 10.3.0.27 Wed Jul 25 17:02:21 1984
vers 2, typ 5, code 1, status 1, AS # 4, id 30151

EGP SENT 10.1.0.52 -> 10.3.0.27 Wed Jul 25 17:02:23 1984
vers 2, typ 5, code 0, status 1, AS # 4, id 2

EGP RECV 10.3.0.27 -> 10.1.0.52 Wed Jul 25 17:02:23 1984
vers 2, typ 5, code 1, status 1, AS # 256, id 2

EGP RECV 10.3.0.27 -> 10.1.0.52 Wed Jul 25 17:02:47 1984
vers 2, typ 5, code 0, status 1, AS # 256, id 30155

EGP SENT 10.1.0.52 -> 10.3.0.27 Wed Jul 25 17:02:47 1984
vers 2, typ 5, code 1, status 1, AS # 4, id 30155

EGP SENT 10.1.0.52 -> 10.3.0.27 Wed Jul 25 17:02:55 1984
vers 2, typ 5, code 0, status 1, AS # 4, id 2

EGP RECV 10.3.0.27 -> 10.1.0.52 Wed Jul 25 17:02:55 1984
vers 2, typ 5, code 1, status 1, AS # 256, id 2

EGP RECV 10.3.0.27 -> 10.1.0.52 Wed Jul 25 17:03:21 1984
vers 2, typ 5, code 0, status 1, AS # 256, id 30160

EGP SENT 10.1.0.52 -> 10.3.0.27 Wed Jul 25 17:03:21 1984
vers 2, typ 5, code 1, status 1, AS # 4, id 30160

EGP SENT 10.1.0.52 -> 10.3.0.27 Wed Jul 25 17:03:27 1984
vers 2, typ 5, code 0, status 1, AS # 4, id 2

EGP RECV 10.3.0.27 -> 10.1.0.52 Wed Jul 25 17:03:27 1984
vers 2, typ 5, code 1, status 1, AS # 256, id 2

EGP RECV 10.3.0.27 -> 10.1.0.52 Wed Jul 25 17:03:47 1984
vers 2, typ 2, code 0, status 0, AS # 256, id 30164, src net 10.0.0.0

EGP SENT 10.1.0.52 -> 10.3.0.27 Wed Jul 25 17:03:47 1984
vers 2, typ 1, code 0, status 1, AS # 4, id 30164, src net 10.0.0.0, # int 1, # ext 0
1 0 52 2 0 1 128 9 1 1 192 5 19
```

```
EGP SENT 10.1.0.52 -> 10.3.0.27 Wed Jul 25 17:03:59 1984
vers 2, typ 2, code 0, status 1, AS # 4, id 3, src net 10.0.0.0

EGP RECV 10.3.0.27 -> 10.1.0.52 Wed Jul 25 17:03:59 1984
vers 2, typ 1, code 0, status 1, AS # 256, id 3, src net 10.0.0.0, # int 26, # ext 1
3 0 27 2 0 1 128 9 2 1 192 5 19 2 0 5 4 0 1 192 1 2 1 3 10 192 1 7 192 1 4 2 1 128 2 255 1 16
5 0 5 3 0 1 26 1 7 24 192 5 9 128 44 6 128 21 192 5 38 128 25 2 15 128 49 192 5 35 192 5 21 128 20
 192 5 25 192 5 22 128 3 128 8 192 5 15 192 5 27 192 5 26 192 5 52 192 5 66 192 5 54 192 5 24
1 0 20 4 0 2 4 7 1 4 192 5 28 192 5 29 32 128 16 2 2 192 5 30 18 3 3 128 31 192 10 41 128 52
2 0 22 1 2 1 192 5 89 2 0 25 2 1 1 192 5 31 2 1 192 5 56 1 0 28 1 0 1 192 5 18
2 0 37 2 0 1 128 10 2 1 192 5 48 1 0 49 2 0 2 192 1 3 128 11 2 1 192 5 51 4 0 51 1 2 2 192 5 14 192 5 64
3 0 72 2 0 1 8 2 1 192 5 88 0 0 94 1 0 1 192 5 2 0 0 4 1 1 1 192 5 12 2 0 9 1 1 1 128 36
1 0 11 1 1 1 36 0 0 15 1 1 1 192 5 37 0 0 25 1 1 1 192 5 11 7 0 49 1 1 1 192 5 58 1 0 54 1 1 1 192 5 7
2 0 78 1 1 1 128 32 3 0 89 1 1 1 192 5 43 3 0 91 1 1 1 192 5 8 3 0 96 1 1 1 192 5 36 0 0 68 1 2 1 192 5 6
5 0 51 1 0 1 128 18 5 0 63 2 0 1 14 2 1 128 42 0 0 111 1 0 2 128 5 128 4

EGP RECV 10.3.0.27 -> 10.1.0.52 Wed Jul 25 17:04:21 1984
vers 2, typ 5, code 0, status 1, AS # 256, id 30169

EGP SENT 10.1.0.52 -> 10.3.0.27 Wed Jul 25 17:04:21 1984
vers 2, typ 5, code 1, status 1, AS # 4, id 30169
 .
 .
 .

OLD: DELETE EXT dst 192.5.46.0, router 10.2.0.25, metric 1, flags UP|GATEWAY, state
OLD: DELETE EXT dst 128.39.0.0, router 10.2.0.25, metric 1, flags UP|GATEWAY, state
rt_time: above old routes deleted Wed Jul 25 17:09:19 1984
 .
 .
 . (full packet traces have been omitted in the following)
 .

egpacq: cease from 10.3.0.27, reason 1, Mon Jul 30 09:13:55 1984

egpstngh: acquired 10.2.0.25 Mon Jul 30 09:14:27 1984

egpacq: cease from 10.2.0.25, reason 1, Mon Jul 30 09:16:36 1984

egpstngh: acquired 10.3.0.27 Mon Jul 30 09:18:11 1984

egp_send: sendto: 10.3.0.27 : Host is down
egp_send: sendto: 10.3.0.27 : Host is down
egpjob: 10.3.0.27 unreachable Mon Jul 30 09:26:11 1984

egp_send: sendto: 10.3.0.27 : Host is down
egpcease: cease to 10.3.0.27, reason 0, Mon Jul 30 09:26:11 1984

egpstngh: acquired 10.2.0.25 Mon Jul 30 09:26:11 1984
```

```
egp_send: sendto: 10.3.0.27 : Host is down
egp_send: sendto: 10.3.0.27 : Host is down
egp_send: sendto: 10.3.0.27 : Host is down
.
.
.

EGP SENT 10.1.0.52 -> 10.2.0.25 Mon Jul 30 09:57:03 1984
vers 2, typ 3, code 3, status 5, AS # 4, id 15
egpcease: cease to 10.2.0.25, reason 5, Mon Jul 30 09:57:03 1984

EGP RECV 10.2.0.25 -> 10.1.0.52 Mon Jul 30 09:57:03 1984
vers 2, typ 3, code 4, status 5, AS # 256, id 15
ADD DEFAULT dst 0.0.0.0, router 10.2.0.22, metric 254, flags UP|GATEWAY, state PASSIVE

Exit egpup at Mon Jul 30 09:57:03 1984
```

REFERENCES

[Berkeley 83] "UNIX Programmer's Manual", Vol. 1, 4.2 Berkeley Software Distribution, University of California, Berkeley.

[Kirton 84] Kirton, P.A., "EGP Gateway Under Berkeley UNIX 4.2", Network Information Center RFC 911, to be published.

[Mills 83] Mills, D.L., "EGP Models and Self-Organizing Systems", Message to EGP-PEOPLE@BBN-UNIX, November 1983.

[Mills 84a] Mills, D.L., "Exterior Gateway Protocol Formal Specification", Network Information Center RFC 904, April 1984.

[Mills 84b] Mills, D.L., "Revised EGP Model Clarified and Discussed", Message to EGP-PEOPLE@BBN-UNIX, May 1984.

[Postel 84] Postel, J., "Exterior Gateway Protocol Implementation Schedule", Network Information Center RFC 890, February 1984.

[Rose 84] Rose, M.T., "Low-Tech Connection into the ARPA-Internet: The Raw-Packet Split Gateway", Department of Information and Computer Science, University of California, Irvine, Technical Report 216, February 1984.

[Rosen 82] Rosen, E.C., "Exterior Gateway Protocol", Network Information Center RFC 827, October 1982.

[Seamonson & Rosen 84]
Seamonson, L.J., and E.C. Rosen, "Stub Exterior Gateway Protocol", Network Information Center RFC 888, January 84.

[Xerox 81] "Internet Transport Protocols", Xerox System Integration Standard XSIS 028112, December 1981.